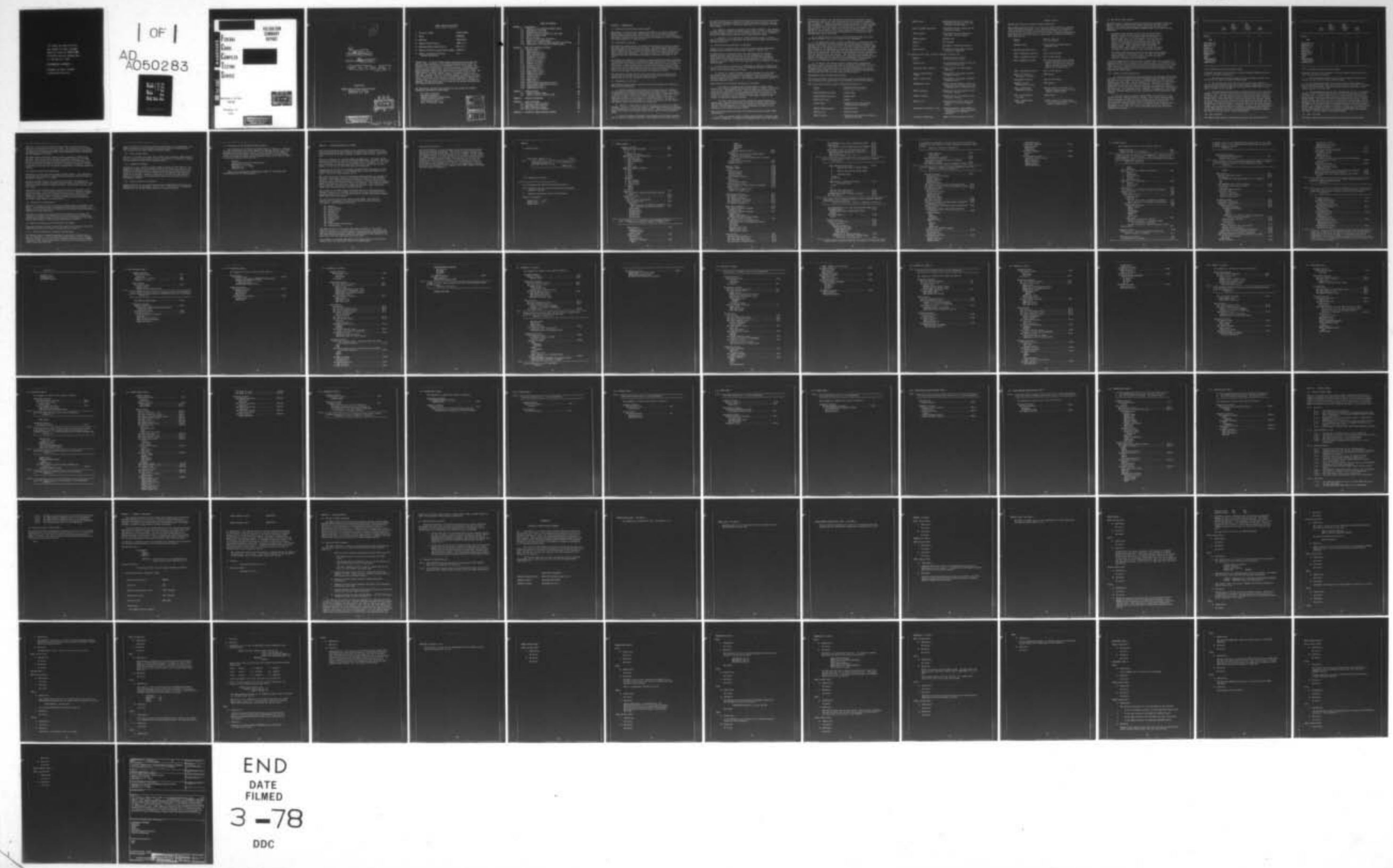
AD-A050 283 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
COBOL COMPILER VALIDATION SUMMARY REPORT.(U)
FEB 78
UNCLASSIFIED CCVS74-VSR285 NL

| OF |
AD
A050283



END
DATE
FILMED
3-78
DDC

VALIDATION SUMMARY REPORT

FEDERAL COBOL COMPILER TESTING SERVICE

Department of the Navy
(ADPESO)

Washington, D.C.
20376

10

6

COBOL COMPILER VALIDATION SUMMARY REPORT.

VALIDATION NUMBER 14 CCVS74-VSR285

11 15 Feb 78 12 83p.

Prepared By:

FEDERAL COBOL COMPILER TESTING SERVICE
DEPARTMENT OF THE NAVY
WASHINGTON, D.C. 20376

408 438 Gu

COBOL COMPILER VALIDATION

1.	Validation Number	CCVS74-VSR285
2.	Vendor	BURROUGHS
3.	Mainframe	B6700/B7700
4.	Compiler Identification	COBOL II.9.1
5.	Operating System Identification	MCP II.9.1
6.	Compiler Validation System Version Number	CCVS74 2.0
7.	Federal Information Processing Standard Publication	21-1

For information concerning this compiler you can contact the vendor's designated representative named below:

Mr. James S. Gutherie
Burroughs Corporation
Federal and Special Systems Group
Standard Products Division
P. O. Box 517 GVL #1
Paoli, Pennsylvania 19301

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

1.1 Purpose of the Validation Summary Report

The purpose of the Validation Summary Report (VSR) is to identify individual COBOL language elements whose implementation does not conform to American National Standard Programming Language COBOL, X3.23-1974, and to Federal Standard COBOL as adopted from the American National Standard by Federal Information Processing Standard 21-1 (FIPS PUB 21-1).

1.2 Preparation of the VSR

The Validation Summary Report is prepared by analyzing the results of running the COBOL Compiler Validation System (CCVS). The COBOL Compiler Validation System consists of audit routines containing features of Federal Standard COBOL, their related data, and an executive routine (VP-routine) which prepares the audit routines for compilation. Each audit routine is a COBOL program which includes many tests and supporting procedures indicating the result of the tests.

The testing of a compiler in a particular hardware/operating system environment is accomplished by compiling and executing each audit routine. The report produced by each routine tells whether the compiler passed or failed the tests in the routine. If the compiler rejects some language elements by terminating compilation, giving fatal diagnostic messages, or terminating execution abnormally, then the test containing the code the compiler was unable to process is deleted and the audit routine compilation and execution repeated.

The compilation listings and the output reports of the audit routines constitute the raw data from which the members of the Federal COBOL Compiler Testing Service produce a Validation Summary Report.

1.3 Organization of the VSR

The Validation Summary Report is made up of several sections the contents of which are described below.

a. Section 2 summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System. Section 2 is subdivided into a subsection representing each level of each module defined in American National Standard Programming Language COBOL, X3.23-1974.. Each subsection contains a list of all of the language elements which must be implemented in order to claim support of that level/module. The list of language elements will be annotated to include a description of both syntax and semantic errors detected during the validation.

b. Section 3 - FIPS PUB 21-1 defines four Federal levels of the COBOL Standard. Section 3.1 of the VSR lists the discrepancies described in Section 2 by the Federal level in which the problem occurs. Section 3.2 lists discrepancies for the Report Writer Module, which is not a part of Federal Standard COBOL.

c. Section 4 contains information which describes the software environment in which the compiler was tested. This includes the name and version of the operating system; the implementor-names which were used in the Environment Division of the programs comprising the CCVS; the options used with the compiler; and if applicable, information regarding the use of compiler optimization features.

d. Section 5 contains the results of the ASCII validation. The purpose of these tests is to ascertain whether magnetic tapes written in ASCII code and with ANSI standard labels, and card decks with ASCII code, can be transported between the system being validated and a foreign computer system.

e. Appendix A is the Validation Summary Working Document, a working paper resulting from the compilation and execution of the CCVS, and from which the VSR is derived.

1.4 Abstract Covering Compliance to ANS COBOL

Definition of an Implementation of American National Standard Programming Language COBOL (excerpts from X3.23-1974, Chapter 1, Section 1.5).

An implementation is defined to meet the requirements of the American National Standard COBOL specification if that implementation includes a fully implemented specified level of each of the functional processing modules and of the Nucleus as defined in this Standard. It follows from this that, in order to [SIC] meet the requirements of this Standard, an implementation must:

a. Not require the inclusion of substitute or additional language elements in the source program, in order to accomplish any part of the function of any of the standard language elements.

b. Accept all standard language elements contained in a given level of a module which is specified as being included in the implementation, except as specifically exempted (as pertaining to specific hardware components for which support is not claimed). See "Elements that Pertain to Specific Hardware Components" below.

These points are of particular pertinence in two areas:

(1) There are throughout the American National Standard COBOL specification certain language elements whose syntax, or effect, is specified to be, in part, implementor-defined. While the implementor specifies the constraints on that portion of each element's syntax or rules that is indicated in this Standard to be implementor-defined, such constraints may not include any requirement for the inclusion in the source program of substitute or additional language elements.

(2) When a function is provided outside the source program that accomplishes a function specified by any particular standard COBOL element, then the implementation must not require, except for Environment Division elements, the specification of that external function in place of or in addition to that standard language element:

The following qualifications apply to the American National Standard COBOL specification:

a. There are certain language elements which pertain to specific types of hardware components. In order for an implementation to meet the requirements of this standard, the implementor must specify the minimum hardware configuration required for that implementation and the hardware components that it supports. Further, when support is thus claimed for a specific hardware component, all standard language elements that pertain to that component must be implemented if the module in which they appear is included in the implementation. Language elements that pertain to specific hardware components for which support is not claimed, need not be implemented. However, the absence of such elements from an implementation of American National Standard COBOL must be specified.

b. An implementation of American National Standard COBOL may include the ENTER statement or not, at the option of the implementor.

c. An implementation that includes, in addition to a specified level of each of the functional processing modules and of the Nucleus, elements or functions that either are not defined in the American National Standard COBOL specification or are defined in a given level of a standard module not otherwise included in the implementation, meets the requirements of this Standard. This is true even though it may imply the extension of the list of reserved words by the implementor, and prevent proper compilation of some programs that meet the requirements of this Standard. The implementor must specify any optional language (language not defined in a specified level but defined elsewhere in the Standard) or extensions (language elements or functions not defined in this Standard) that are included in the implementation.

d. In general, the American National Standard COBOL specification specifies no upper limit on such things as the number of statements in a program, the number of operands permitted in certain statements, etc. It is recognized that these limits will vary from one implementation of American National Standard COBOL to another and may prevent the proper compilation of some programs that meet the requirements of this standard.

IMPLEMENTOR-DEFINED LANGUAGE SPECIFICATIONS

The language elements in the following lists depend on implementor definitions to complete the specificaton of the syntax or rules for the elements.

The elements whose syntax is partly implementor-defined are:

Element	Implementor-Defined Aspect
SOURCE-COMPUTER paragraph	computer-name
OBJECT-COMPUTER paragraph	computer-name
MEMORY SIZE clause	integer
alphabet-name	implementor-name; whether implementor-names are provided.
SPECIAL-NAMES paragraph	implementor-name
ASSIGN clause	implementor-name
VALUE OF clause	implementor-name; whether implementor-names are provided.

RERUN clause	implementor-name and the form; the implementor provides at least one of seven specified forms.
CALL and CANCEL statements	relationship between operand and the referenced program.
COPY statement	relationship between library-name text-name, and the library.
ENTER statement	language-name
Margin R	The location.
Area B	The number of character positions.
Qualification	The number of qualifiers; at least five must be supported.

The elements whose effect is partly implementor-defined are:

Element	Implementor-Defined Aspect
alphabet-name	The correspondence between native and foreign character sets.
implementor-name switches	Whether setting can change during execution.
USAGE IS COMPUTATIONAL clause	Representation and whether automatic alignment occurs.
USAGE IS INDEX clause	Representation and whether automatic alignment occurs.
SYNCHRONIZED clause	Whether implicit FILLER positions are generated; their effect on the size of group items and redefining items.
ACCEPT statement	Maximum size of one transfer of data in Level 1 Nucleus.
DISPLAY statement	Maximum size of one transfer of data in Level 1 Nucleus.
Numeric test	Representation of valid sign in the absence of the SIGN IS SEPARATE clause.
Comparison of nonnumeric items	Collating sequence, where NATIVE or implementor-name collating sequence is implicitly or explicitly specified.
Arithmetic expressions	Number of places carried for inter-

mediate results.

Elements That Pertain to Specific Hardware Components

The standard language elements in the list that follows pertain to specific types of hardware components. These language elements must be implemented in an implementation of American National Standard COBOL when support is claimed, by the implementor, for the specific types of hardware components to which they pertain, and the module in which they are defined is included in that implementation.

Element	Hardware Component
CODE-SET clause	Device capable of supporting the specified code.
MULTIPLE FILE TAPE clause	Reel
CLOSE...REEL/UNIT statement	Reel or mass storage
CLOSE...NO REWIND statement	Reel or mass storage
OPEN...REVERSED statement	Reel with the capability of making records available in the reversed order; mass-storage with the capability of making records available in the reversed order.
OPEN...NO REWIND statement	Reel or mass storage
OPEN...I-O statement (Sequential I-O only)	Mass storage
OPEN EXTEND statement	Reel or mass storage
REWRITE statement (Sequential I-O only)	Mass storage
SEND...BEFORE/AFTER ADVANCING statement	Devices capable of vertical positioning; devices capable of action based on mnemonic-names.
USE...I-O (Sequential I-O only)	Mass storage
WRITE...BEFORE/AFTER ADVANCING	Devices capable of vertical positioning; devices capable of action based on mnemonic-name.

1.5 The Federal COBOL Standard

The COBOL compiler validation results enclosed in this document reflect the degree to which the subject COBOL compiler implements the Federal COBOL Standard. The Federal COBOL Standard is essentially the same as the American National Standard Programming Language COBOL, X3.23-1974, with two exceptions:

> The Federal COBOL Standard defines 4 levels and the ANSI Standard defines only the minimum COBOL implementation and the full standard. Low and High levels of the Federal COBOL Standard (see 1.5.1) correspond to the above two ANSI levels (minus the Report Writer module). Two additional levels, low-intermediate and high-intermediate have been included in the Federal Standard between the highest and lowest subsets. These additional levels accommodate hardware which cannot support the full standard, but which is capable of implementing more than the minimum standard.
>
> The Federal COBOL Standard states that the Report Writer Module is not mandatory in any Federal level, but that the specifications contained in X3.23-1974 should be used to the extent practical, consistent with requirements.

The Federal COBOL Standard requires that a compiler contain as a minimum the elements specified in at least one of the Federal levels. No restrictions are imposed on the inclusion of selected features from higher levels or even unique vendor extensions. Compatibility amoung various implementations of a given level containing additional features must be controlled by management imposed standards and restrictions.

1.5.1 Federal Standard COBOL Levels

a. Federal Standard COBOL specifications are the language specifications contained in American National Standard Programming Language COBOL, X3.23-1974. For purposes of the Federal Standard, the modules defined in X3.23-1974 are combined into four levels. Not all computers are large enough to accommodate a COBOL compiler containing the full ANSI Standard. Therefore, the Federal Government requires that all compilers acquired by its agencies contain as a minimum one of the four Federal levels, depending on machine size, configuration and user needs. The knowledge that all computers will support at least one of these four subsets simplifies the task of developing machine-independent COBOL programs.

b. The four levels of Federal Standard COBOL are identified as: Low, Low-Intermediate, High-Intermediate, and High. Each Federal Standard COBOL level is composed of either the high or low levels of the nucleus and ten of the eleven Functional Processing Modules (FPMs) defined in X3.23-1974. The four Federal Standard COBOL levels are reflected in the following table. The numbers in the table refer to the level within the FPM or nucleus as designated in X3.23-1974, and a dash in the table denotes that the corresponding FPM is omitted.

	Low Level	Low Inter-mediate Level	High Inter-mediate Level	High Level
NUCLEUS	1	1	2	2
FPMs				
TABLE HANDLING	1	1	2	2
SEQUENTIAL I-O	1	1	2	2
RELATIVE I-O	-	1	2	2
INDEXED I-O	-	-	-	2
SORT-MERGE	-	-	1	2
REPORT WRITER	-	-	-	-
SEGMENTATION	-	1	1	2
LIBRARY	-	1	1	2
DEBUG	-	1	2	2
INTER-PROGRAM COMMUNICATION	-	1	2	2
COMMUNICATION	-	-	2	2

1.5.2 Conformance to Federal Standard COBOL

A compiler implemented in conformance to Federal Standard COBOL must meet at least the following requirements.

a. The implementation must include all of the language elements of at least one of the levels of Federal Standard COBOL.

b. The implementation must meet all of the requirements defined in American National Standard COBOL, X3.23-1974, Section I, paragraph 1.5, Definition of An Implementation of American National Standard COBOL which is provided in section 1.4 of this VSR.

c. The implementation must provide a facility for the user to optionally specify a level of Federal Standard COBOL for monitoring his source program at compile time. The monitoring will be an analysis of the syntax used in a source program against the syntax included in the specified level of Federal Standard COBOL. Any syntax used in the source program that does not conform to that allowed by the user selected level of Federal Standard COBOL will be diagnosed. The syntax diagnosed as not conforming to the specified level will be identified to the user through a diagnostic message on the source program listing. The diagnostic message will contain, at least: (1) The identification of the source program line number in which the nonconforming syntax occurs, (2) the identification of the level of Federal Standard COBOL that supports the syntax or that the syntax is nonstandard COBOL.

1.6. Use of the VSR

The Federal COBOL Compiler Testing Service may make full and free public

	Low Level	Low Inter-mediate Level	High Inter-mediate Level	High Level
NUCLEUS	1	1	2	2
FPMs				
TABLE HANDLING	1	1	2	2
SEQUENTIAL I-O	1	1	2	2
RELATIVE I-O	-	1	2	2
INDEXED I-O	-	-	-	2
SORT-MERGE	-	-	1	2
REPORT WRITER	-	-	-	-
SEGMENTATION	-	1	1	2
LIBRARY	-	1	1	2
DEBUG	-	1	2	2
INTER-PROGRAM COMMUNICATION	-	1	2	2
COMMUNICATION	-	-	2	2

1.5.2 Conformance to Federal Standard COBOL

A compiler implemented in conformance to Federal Standard COBOL must meet at least the following requirements.

a. The implementation must include all of the language elements of at least one of the levels of Federal Standard COBOL.

b. The implementation must meet all of the requirements defined in American National Standard COBOL, X3.23-1974, Section I, paragraph 1.5, Definition of An Implementation of American National Standard COBOL which is provided in section 1.4 of this VSR.

c. The implementation must provide a facility for the user to optionally specify a level of Federal Standard COBOL for monitoring his source program at compile time. The monitoring will be an analysis of the syntax used in a source program against the syntax included in the specified level of Federal Standard COBOL. Any syntax used in the source program that does not conform to that allowed by the user selected level of Federal Standard COBOL will be diagnosed. The syntax diagnosed as not conforming to the specified level will be identified to the user through a diagnostic message on the source program listing. The diagnostic message will contain, at least: (1) The identification of the source program line number in which the nonconforming syntax occurs, (2) the identification of the level of Federal Standard COBOL that supports the syntax or that the syntax is nonstandard COBOL.

1.6. Use of the VSR

The Federal COBOL Compiler Testing Service may make full and free public disclosure of the Validation Summary Report (VSR) in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of the validation are only for the purpose of satisfying United States Government requirements, and apply only to the computer system, operating system release, and compiler version identified in the VSR.

The COBOL Compiler Validation System is used to determine, insofar as is practical, the degree to which the subject compiler conforms to the COBOL Standard. Thus, the VSR is necessarily discretionary and judgmental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in the VSR are accurate or complete. The VSR is not meant to be used for the purpose of publicizing the findings summarized therein.

1.7 Sources of Additional Information

FIPS PUB 21-1 defines the Federal COBOL Language Standard. This publication is available from the Office of ADP Standards Management, National Bureau of Standards, Washington, D. C., 20234.

The detailed COBOL language specifications are given in the publication "American National Standard Programming Language COBOL, X3.23-1974", available from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

An explanation of the COBOL Compiler Validation System is contained in the CCVS User's Guide. This document explains how to run the compiler validation system. The User's Guide and a magnetic tape containing a copy of the CCVS programs are available from the National Technical Information Service, Springfield, Virginia, 22151. (Ordering information can be obtained from the Federal COBOL Compiler Testing Service.)

1.8. Requests for Interpretation

Questions regarding this VSR or the CCVS in general should be forwarded to the FCCTS. If any problem cannot be adequately resolved through the FCCTS, the request for interpretation will be forwarded to the Federal COBOL Interpretation Committee for final resolution.

A brochure describing the validation process including the procedures for requesting a validation and resolution of questions involving interpretation of the current Federal Standard is available from the Department of the Navy, Federal COBOL Compiler Testing Service, Washington, D.C. 20376.

1.9 Modules and Language Elements Excluded from Testing

During an official validation, certain CCVS tests may not be used, and certain facilities provided by the subject compiler may not be tested.

1.9.1 Federal Standard COBOL Approved Interpretations

The National Bureau of Standards published in the Federal Register Vol. 41 No. 179, September 14, 1976, an approved interpretation of Federal Standard COBOL as pertains to the evaluation of arithmetic expressions in the COMPUTE statements. This interpretation states that "size of the intermediate result field is implementor-defined."

Since the results of evaluating arithmetic expressions are not predictable, all COMPUTE statements and IF statements containing arithmetic expressions have been removed from the COBOL Compiler Validation System.

1.9.2 Report Writer Module

FIPS PUB 21-1 excludes the Report Writer Module from the Federal COBOL Standard. However, the Report Writer Module is still tested during a validation if support for that module is claimed by the compiler vendor.

1.9.3 Communication Module

Although it is part of Federal Standard COBOL as defined by FIPS PUB 21-1, the Communication Module is not currently tested in the course of an official validation for two specific reasons. First, a large volume of requests for interpretation on this module have been submitted to the cognizant ANSI committee (X3J4) for resolution. Secondly, facilities for testing were insufficient to determine the validity of the Communication Module test programs during the development of CCVS74.

1.9.4 Vendor Omissions or Extensions

Language elements are not tested which have been legitimately omitted from the implementation by the implementor (refer to 1.4). Additionally, no implementor extensions to the standard COBOL language are tested in any way.

1.10 Timeliness of the Validation Summary Reports

The timeliness of the Validation Summary Report is important. Compilers and their related operating system software are modified several times a year. The Compiler Validation System used to validate compilers is also updated during the life of the system. Therefore to ensure that the latest version of both the vendor's compiler and the Validation System are the latest officially released versions, check with the:

Director
Federal COBOL Compiler Testing Service
Department of the Navy
Washington, D. C. 20376
(202) 697-1247

Please use the Validation Summary Report number of this report when corresponding with the Testing Service.

SECTION 2. DETAILED EVALUATION OF ERRORS.

This section summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System (CCVS). The version of the CCVS used during this validation is shown inside the front cover of the VSR.

Section 2 is made up of a variable number of subsections. The number of subsections is dependent on the Level of Federal COBOL being validated. There will be a subsection for each level of each module which is validated. If the high level of a module is validated then there will be two subsections for that module; one for the low level and one for the high level.

A validation of the low level of Federal Standard COBOL would result in three subsections being present. One for Nucleus level 1, one for Sequential I-O level 1, and one for Table Handling level 1.

Each error or deviation noted in this section makes reference to a program or functional COBOL module contained in Appendix A (Validation Summary Working Document). This reference provides the documented results of an occurrence of errors/deviataions detected during the running of the CCVS using the compiler within the environment identified within this document. The Validation Summary Working Document is presented in sequence by functional module, functional module level and program number as defined below.

Each program in the COBOL Compiler Validation System is identified by a 5-character program name. The name associates the routine with the functional processing module and level of American National Standard Programming Language COBOL tested within the program.

The five character name has the general format XXNMM. The first two characters are alphabetic and identify the functional module tested by the program. The permissable values are:

NC - Nucleus
TH - Table Handling
SQ - Sequential I-O
RL - Relative I-O
IX - Indexed I-O
ST - Sort-Merge
RW - Report Writer
SG - Segmentation
LB - Library
DB - Debug
IC - Inter-Program Communication
CM - Communication

The third character of the audit routine name is either a 1 or 2, and identifies the level of the functional module being tested. Each module and level is represented by several programs. The fourth and fifth characters of the program name are sequence numbers for programs which test features in the same level of the same functional processing module.

As an example, the program name NC210 is the tenth program in the series of routines which test the second level of the Nucleus module.

Description of Section 2.

Each error/deviation is noted by number in the left hand margin opposite the language element in question. This number is used in section 3 to categorize errors by Federal level (See 1.5.1). Inserted directly below the language element is a brief description of the error. To the right of the langauge element is a page reference to X3.23-1974, American National Standard Programming Language COBOL. The reference at the end of the description of the error is to Appendix A which contains the detailed information collected during the validation. The reference is made up of the routine name followed by an A or B (A for compile time or syntax error and B for execution time or semantic error) and a number which makes the error unique in Appendix A.

Example:

```
  2.1 Nucleus Level 1

        .
        .
        .
          Operational symbols: S V P                     II-21
 2.1.9       ------------------------------------------------
             * The scaling character 'P' is not permitted in a
             * PICTURE character-string.
             *                                    (NC101.A.2)
             ------------------------------------------------

        .
        .
        .

   2.2  Sequential I-O Level 1
```

2.1.9 represents the ninth error for Nucleus Level 1

II-21 represents the page in X3.23-1974 where the language element is defined

*   Boxes the description of the error/deviation

NC101.A.2 represents:

Program name	-	NC101
Syntax error	-	A
second error	-	2

2.1 NUCLEUS LEVEL 1

```
      Language Concepts . . . . . . . . . . . . . . . . . .   I-75
        Characters used for words . . . . . . . . . . . . .   I-76
          0, 1, ..., 9
          A, B, ..., Z
          - (hyphen or minus)
        Characters used for punctuation . . . . . . . . . .   I-65
          "  quotation mark
          (  left parenthesis
          )  right parenthesis
          .  period
             space
          =  equal sign
        Characters used in editing. . . . . . . . . . . . .   I-58
          B  space
          0  zero
          +  plus
          -  minus
          CR credit
          DB debit
          Z  zero suppress
          *  check protect
          $  currency sign
          ,  comma
          .  period
          /  stroke
        Separators. . . . . . . . . . . . . . . . . . . . .   I-75
          The separators, semicolon and comma, are not
            allowed . . . . . . . . . . . . . . . . . . . .   II-1
        Character-strings . . . . . . . . . . . . . . . . .   I-76
          COBOL words . . . . . . . . . . . . . . . . . . .   I-76
            Not more than 30 characters
            User-defined words. . . . . . . . . . . . . . .   I-76
              data-name
                Must begin with an alphabetic character       II-1
                Must be unique; may not be qualified. .       II-1
              level-number
              mnemonic-name
              paragraph-name
              program-name
              routine-name
              section-name
2.1.1 ---------------------------------------------------------------------
    * Null SECTIONs are not permitted in the PROCEDURE DIVISION.
    *          (NC114.A.1, NC160.A.1, SG204.A.1, ST201.A.1)
      ---------------------------------------------------------------------

            System-names. . . . . . . . . . . . . . . . . .   I-78
              computer-name
              implementor-name
              language-name
            Reserved words. . . . . . . . . . . . . . . . .   I-79
              Key words
              Optional words
              Figurative constants. . . . . . . . . . . . .   I-80
                ZERO
```

```
        The JUSTIFIED clause (may be abbreviated JUST).   II-16
        Level-number. . . . . . . . . . . . . . . . . .   II-17
          01 through 10 (level numbers must be 2 digits)  II-13
          77. . . . . . . . . . . . . . . . . . . . . .   II-11
        The PICTURE clause (may be abbreviated PIC) . .   II-18
          Character-string may contain 30 characters. .   II-18
          Data characters:  A X 9 . . . . . . . . . . .   II-18
          Operational symbols:  S  V  P . . . . . . . .   II-21
2.1.2 ----------------------------------------------------------------------
    * Data items with PICTURE S9P(17) are not recognized as integers.
    *          (NC114.A.2)
    ----------------------------------------------------------------------

          Fixed insertion characters. . . . . . . . . .   II-21
            0      (may be used only in edited items)
            ,
            B      (may be used only in edited items)
            .
            $      (currency sign)
            + and -
            DB and CR
            /
          Replacement or floating characters. . . . . .   II-21
            $      (currency sign)
            + and -
            Z
            *
          Currency sign substitution. . . . . . . . . .   II-21
          Decimal point substitution. . . . . . . . . .   II-21
        The REDEFINES clause (may not be nested). . . .   II-27
        The SIGN clause . . . . . . . . . . . . . . . .   II-31
2.1.3 ----------------------------------------------------------------------
    * The clause SIGN IS TRAILING SEPARATE or SIGN IS TRAILING SEPARATE
    * CHARACTER are not implemented.
    *          (NC116.A.1. NC160.A.1, NC217.A.1, NC217.B.1, TH111.A.1)
    ----------------------------------------------------------------------

        The SYNCHRONIZED clause (may be abbreviated SYNC) II-33
        The USAGE clause. . . . . . . . . . . . . . . .   II-35
          COMPUTATIONAL (may be abbreviated COMP)
          DISPLAY
        The VALUE clause. . . . . . . . . . . . . . . .   II-36
          literal

      Procedure Division. . . . . . . . . . . . . . . .   I-99
        Conditional expressions . . . . . . . . . . . .   II-41
          Simple condition. . . . . . . . . . . . . . .   II-41
            Relation condition. . . . . . . . . . . . .   II-41
              Relational operators
                [NOT] GREATER THAN
                [NOT] LESS THAN
                [NOT] EQUAL TO
              Comparison of numeric operands. . . . . .   II-42
              Comparison of nonnumeric operands (oper-
                  ands must be of equal size) . . . . .   II-42
2.1.4 ----------------------------------------------------------------------
    * Signed numeric elementary items are not treated as if they are moved
* to elementary alphanumeric items of the same size to strip the
* operational signs before the actual comparison of operands takes
* place.
*         (NC103.B.1, NC114.B.1, NC160.B.1)
----------------------------------------------------------------------
```

```
2.1.5 ----------------------------------------------------------------------
* Leading spaces are truncated from data ACCEPTed from the SPO.
*         (NC109.B.1, NC158.B.1, NC204.B.1)
----------------------------------------------------------------------
```

2.2 NUCLEUS LEVEL 2

```
        All elements of 1 NUC 1,2 are a part of 2 NUC 1,2

        Language Concepts. . . . . . . . . . . . . . . . . . . . . .   I-75
          Characters used for punctuation. . . . . . . . . . . . .   I-65
2.2.1 -------------------------------------------------------------------------
      * Comma and semicolon should be interchangeable.  They are not fully
      * interchangeable in this compiler.  Comma is not allowed in the
      * DATA DIVISION.  Comma is not allowed between Alphabet-name clauses
      * in the SPECIAL-NAMES paragraph.
      *         (NC213.A.1, NC215.A.1)
      -------------------------------------------------------------------------

            ,    comma
            ;    semicolon
          Characters used for arithmetic operations. . . . . . .   I-52
            +    addition
            -    subtraction
            *    multiplication
            /    division
            **   exponentiation
          Characters used in relations . . . . . . . . . . . . .   I-66
            =    equal to
            >    greater than
            <    less than
          Separators . . . . . . . . . . . . . . . . . . . . . .   I-75
            The separators, semicolon and comma, are allowed . .   II-1
          Character-strings. . . . . . . . . . . . . . . . . . .   I-76
            COBOL words. . . . . . . . . . . . . . . . . . . . .   I-76
              User-defined words . . . . . . . . . . . . . . . .   I-76
                condition-name
                data-name
                  Need not begin with an alphabetic character. .   II-1
                  May be qualified if necessary for uniqueness .   II-1
              Reserved words . . . . . . . . . . . . . . . . . .   I-79
                Figurative constants . . . . . . . . . . . . . .   I-80
                  ZEROS; ZEROES
                  SPACES
                  HIGH-VALUES
                  LOW-VALUES
                  QUOTES
                  ALL literal
                Connectives . . . . . . . . . . . . . . . . . .  I-79
                  Qualifier connectives:  OF, IN
                  Series connectives:  , (separator comma)
                      and ; (separator semicolon)
                  Logical connectives:  AND, OR, AND NOT, OR NOT
          Qualification . . . . . . . . . . . . . . . . . . . .  I-87

        Reference format . . . . . . . . . . . . . . . . . . . .   I-105
          Continuation of lines (continuation of words and
              numeric literals is allowed) . . . . . . . . . . .   II-1

        Identification Division. . . . . . . . . . . . . . . . .   I-94
          The DATE-COMPILED paragraph. . . . . . . . . . . . . .   II-4
2.2.2 -------------------------------------------------------------------------
```

```
* Comment entries in the DATE-COMPILED paragraph that are not formal
* comment lines, are not replaced by the current date on which the
* source program is compiled.
*         (NC203.A.1)
-----------------------------------------------------------------------
```

```
2.2.3 -----------------------------------------------------------------------
* The figurative constants LOW-VALUE, HIGH-VALUE, QUOTE, and SPACES
* are not allowed in literals in the Alphabet-name clause.
*         (NC214.A.1)
-----------------------------------------------------------------------
```

```
          TO identifier series
          GIVING identifier series
          CORRESPONDING phrase
        The ALTER statement. . . . . . . . . . . . . . . . . . . .   II-57
          The series option is allowed
        The COMPUTE statement. . . . . . . . . . . . . . . . . . .   II-58
          identifier series
          ROUNDED phrase
          SIZE ERROR phrase
        The DISPLAY statement (no restrictions on the number
            of transfers of data). . . . . . . . . . . . . . . . .   II-59
          UPON phrase
        The DIVIDE statement . . . . . . . . . . . . . . . . . . .   II-61
          INTO identifier series
          GIVING identifier series
          REMAINDER phrase
        The GO TO statement (procedure-name may be omitted). .   II-65
        The IF statement (nested statements) . . . . . . . . . .   II-66
        The INSPECT statement (multi-character data items) . .   II-68
          series
2.2.4 ---------------------------------------------------------------------
    * Incorrect totals are given for the INSPECT TALLYING statement with
    * multiple BEFORE and AFTER phrases.
    *         (NC216.B.1)
      ---------------------------------------------------------------------

2.2.5 ---------------------------------------------------------------------
    * The operational sign in an INSPECT TALLYING statement is not ignored
    * when using signed numeric data as the field that is inspected.
    *         (NC216.B.2)
      ---------------------------------------------------------------------

        The MOVE statement . . . . . . . . . . . . . . . . . . . .   II-74
          CORRESPONDING phrase
        The MULTIPLY statement . . . . . . . . . . . . . . . . . .   II-77
          BY identifier series
          GIVING identifier series
        The PERFORM statement. . . . . . . . . . . . . . . . . . .   II-78
          UNTIL phrase
          VARYING phrase
        The STRING statement . . . . . . . . . . . . . . . . . . .   II-86
          DELIMITED series
          POINTER phrase
          ON OVERFLOW phrase
        The SUBTRACT statement . . . . . . . . . . . . . . . . . .   II-89
          FROM identifier series
          GIVING identifier series
          CORRESPONDING phrase
        The UNSTRING statement . . . . . . . . . . . . . . . . . .   II-91
          DELIMITED BY phrase
2.2.6 ---------------------------------------------------------------------
    * There is a problem with the UNSTRING statement when the DELIMITED BY
    * ALL ZERO, or ALL "0" or ALL data-name (with zero value) phrases are
    * used with the DELIMITER IN phrase. The receiving field in the
    * DELIMITER IN phrase should contain a single left justified zero with
    * the remainder of the field blank filled when one or a contiguous string
    * of the delimiter is found.
*             (NC218.B.1)
---------------------------------------------------------------------------

        POINTER phrase
        TALLYING phrase
        ON OVERFLOW phrase
```

2.3 TABLE HANDLING LEVEL 1

```
        Language Concepts
          User-defined words. . . . . . . . . . . . . . . .   I-76
            index-name
          Subscripting - 3 levels . . . . . . . . . . . . .   I-89
          Indexing - 3 levels . . . . . . . . . . . . . . .   I-89

        Data Division
          The OCCURS clause . . . . . . . . . . . . . . . .   III-2
            integer TIMES
            INDEXED BY index-name series
2.3.1 -----------------------------------------------------------------
    * If an INDEXED BY phrase appears at the group level, every elementary
    * level within the group requires an INDEXED BY phrase in this compiler.
    *         (TH109.A.1)
      -----------------------------------------------------------------

          The USAGE IS INDEX clause . . . . . . . . . . . .   III-5

        Procedure Division
          Relation conditions . . . . . . . . . . . . . . .   III-6
            Comparisons involving index-names and/or
              index data items
          Overlapping operands. . . . . . . . . . . . . . .   III-6
          The SET statement . . . . . . . . . . . . . . . .   III-11
            index-name/identifier series
            index-name
            UP BY identifier/integer
            DOWN BY identifier/integer
            index-name series
```

2.4 TABLE HANDLING LEVEL 2

All elements of 1 TBL 1,2 are a part of 2 TBL 1,2

2.5 SEQUENTIAL I-O LEVEL 1

```
        EXCEPTION/ERROR PROCEDURE
           ON file-name
           ON INPUT
           ON OUTPUT
           ON I-O
     The WRITE statement . . . . . . . . . . . . . . . . . . . . IV-34
        FROM identifier
        BEFORE/AFTER integer LINES
2.5.1 ------------------------------------------------------------------------
      * There is a line spacing problem with WRITE AFTER ADVANCING integer
      * LINES clause.
      *          (SQ101.B.1, SQ151.B.1)
      ------------------------------------------------------------------------

        BEFORE/AFTER PAGE
```

2.6 SEQUENTIAL I-O LEVEL 2

All elements of 1 SEQ 1,2 are a part of 2 SEQ 1,2

```
Language Concepts
   Special register . . . . . . . . . . . . . . . . . . . .    I-80
      LINAGE-COUNTER. . . . . . . . . . . . . . . . . . . .    IV-3

Environment Division
   The FILE-CONTROL paragraph . . . . . . . . . . . . . . .    IV-4
   The file control entry . . . . . . . . . . . . . . . . .    IV-4
      SELECT clause
         OPTIONAL phrase
      RESERVE integer AREA(S) clause
   The I-O-CONTROL paragraph. . . . . . . . . . . . . . . .    IV-6
      SAME RECORD AREA clause
      SAME RECORD AREA series
      MULTIPLE FILE TAPE clause

Data Division
   The file description entry. . . . . . . . . . . . . . . .   IV-10
   The BLOCK CONTAINS clause . . . . . . . . . . . . . . . .   IV-11
      integer-1 TO integer-2 RECORDS
      integer-1 TO integer-2 CHARACTERS
   The LINAGE clause . . . . . . . . . . . . . . . . . . . .   IV-15
```

2.6.1 --

* Extra blank lines appear after the last line of a logical page
* and the first line on the next logical page.
*   (SO213.B.1, SO214.B.1)

```
      FOOTING phrase
      TOP phrase
      BOTTOM phrase
   The VALUE OF clause . . . . . . . . . . . . . . . . . . .   IV-19
      implementor-name IS data-name
      implementor-name IS data-name series

Procedure Division
   The CLOSE statement . . . . . . . . . . . . . . . . . . .   IV-20
      NO REWIND, REMOVAL, or LOCK
      file-name series
   The OPEN statement. . . . . . . . . . . . . . . . . . . .   IV-24
      INPUT
         REVERSED
         NO REWIND
      OUTPUT
         NO REWIND
      EXTEND
      file-name series
      INPUT, OUTPUT, I-O, and EXTEND series
   The USE statement . . . . . . . . . . . . . . . . . . . .   IV-32
      EXCEPTION/ERROR PROCEDURE ON file-name series
      EXCEPTION/ERROR PROCEDURE ON EXTEND
```

2.6.2 --

* The EXTEND option is not implemented.
*   (SO218.A.1)

2.7 RELATIVE I-O LEVEL 1

2.7.1 --
* The Relative I-O Module Level 1 is not implemented.

```
Language Concepts
   User-defined words. . . . . . . . . . . . . . . . . . . .   I-76
      file-name
      record-name
   I-O status. . . . . . . . . . . . . . . . . . . . . . . .   V-2

Environment Division
   The FILE-CONTROL paragraph. . . . . . . . . . . . . . . .   V-5
   The file control entry. . . . . . . . . . . . . . . . . .   V-5
      SELECT clause
      ASSIGN TO implementor-name clause
      ORGANIZATION IS RELATIVE clause
      ACCESS MODE clause
         SEQUENTIAL
         RANDOM
      FILE STATUS clause
   The I-O-CONTROL paragraph . . . . . . . . . . . . . . . .   V-7
      RERUN clause
      SAME AREA clause
      SAME AREA series

Data Division
   File Section. . . . . . . . . . . . . . . . . . . . . . .   V-10
   The file description entry. . . . . . . . . . . . . . . .   V-11
   The record description entry. . . . . . . . . . . . . . .   V-10
   The BLOCK CONTAINS clause . . . . . . . . . . . . . . . .   V-12
      integer CHARACTERS
      integer RECORDS
   The DATA RECORDS clause . . . . . . . . . . . . . . . . .   V-13
      data-name
      data-name series
   The LABEL RECORDS clause. . . . . . . . . . . . . . . . .   V-14
      STANDARD
      OMITTED
   The RECORD CONTAINS clause. . . . . . . . . . . . . . . .   V-15
      integer-1 TO integer-2 CHARACTERS
   The VALUE OF clause . . . . . . . . . . . . . . . . . . .   V-16
      implementor-name IS literal
      implementor-name IS literal series

Procedure Division
   The CLOSE statement . . . . . . . . . . . . . . . . . . .   V-17
      WITH LOCK
      file-name series
   The DELETE statement. . . . . . . . . . . . . . . . . . .   V-19
      INVALID KEY phrase
   The OPEN statement. . . . . . . . . . . . . . . . . . . .   V-20
      INPUT
      OUTPUT
      I-O
      file-name series
```

2.8 RELATIVE I-O LEVEL 2

2.8.1 --

* The Relative I-O Module Level 2 is not implemented.

All elements of 1 REL 0,2 are a part of 2 REL 0,2

```
Environment Division
   The FILE-CONTROL paragraph. . . . . . . . . . . . . . . . . .  V-5
   The file control entry. . . . . . . . . . . . . . . . . . . .  V-5
      SELECT clause
      RESERVE integer AREA(S) clause
      ACCESS MODE IS DYNAMIC clause
   The I-O-CONTROL paragraph . . . . . . . . . . . . . . . . . .  V-7
      SAME RECORD AREA
      SAME RECORD AREA entries

Data Division
   The file description entry. . . . . . . . . . . . . . . . . .  V-11
   The BLOCK CONTAINS clause . . . . . . . . . . . . . . . . . .  V-12
      integer-1 TO integer-2 RECORDS
      integer-1 TO integer-2 CHARACTERS
   The VALUE OF clause . . . . . . . . . . . . . . . . . . . . .  V-16
      implementor-name IS data-name
      implementor-name IS data-name entries

Procedure Division
   The READ statement. . . . . . . . . . . . . . . . . . . . . .  V-23
      NEXT RECORD
   The START statement . . . . . . . . . . . . . . . . . . . . .  V-28
      KEY IS phrase
      INVALID KEY phrase
   The USE statement . . . . . . . . . . . . . . . . . . . . . .  V-30
      EXCEPTION/ERROR PROCEDURE
        ON file-name series
```

2.9 INDEXED I-O LEVEL 1

2.10 INDEXED I-O LEVEL 2

All elements of 1 INX 0,2 are a part of 2 INX 0,2

Environment Division
- The FILE-CONTROL paragraph VI-5
- The file control entry VI-5
  - SELECT clause
  - RESERVE integer AREA(S) clause
  - ACCESS MODE IS DYNAMIC clause
  - ALTERNATE RECORD KEY clause
    - WITH DUPLICATES phrase

2.10.1 --

* The ALTERNATE RECORD KEY clause and the WITH DUPLICATES phrase
* are not implemented.
* (IX205.A.1)

- The I-O-CONTROL paragraph. VI-8
  - SAME RECORD clause
  - SAME RECORD AREA series

Data Division
- The file description entry VI-12
- The BLOCK CONTAINS clause. VI-13
  - integer-1 TO integer-2 RECORDS
  - integer-1 TO integer-2 CHARACTERS
- The VALUE OF clause. VI-17
  - implementor-name IS data-name
  - implementor-name IS data-name series

Procedure Division
- The READ statement VI-24
  - KEY IS phrase
  - NEXT RECORD
- The START statement. VI-30
  - KEY IS phrase
  - INVALID KEY phrase
- The USE statement. VI-32
  - EXCEPTION/ERROR PROCEDURE
    - ON file-name series

2.11 SORT-MERGE LEVEL 1

2.12 SORT-MERGE LEVEL 2

All elements of 1 SRT 0,2 are a part of 2 SRT 0,2

```
Environment Division
   The FILE-CONTROL paragraph. . . . . . . . . . . . . . . .  VII-2
   The file control entry. . . . . . . . . . . . . . . . . .  VII-2
      SELECT clause
   The I-O-CONTROL paragraph . . . . . . . . . . . . . . . .  VII-3
      SAME RECORD AREA clause
      SAME SORT/SORT-MERGE AREA clause
```

2.12.1
```
---------------------------------------------------------------------
* The SAME SORT-MERGE AREA clause is not implemented.
*         (ST208.A.1)
---------------------------------------------------------------------
```

```
      SAME series

Procedure Division
   The MERGE statement . . . . . . . . . . . . . . . . . . .  VII-8
```

2.12.2
```
----------------------------------------------------------------------
* The same sort or merge file name cannot be used for both sorting
* and merging in the same program because of constraints on the
* SELECT statement; e.g., ASSIGN TO SORT DISK vs. ASSIGN TO MERGE DISK.
*         (ST214.A.1)
----------------------------------------------------------------------
```

```
      KEY data-name
         data-name series
      ASCENDING series
      DESCENDING series
      mixed ASCENDING/DESCENDING
      COLLATING SEQUENCE phrase
```

2.12.3
```
---------------------------------------------------------------------
* The MERGE COLLATING SEQUENCE phrase is not implemented.
*          (ST208.A.4)
---------------------------------------------------------------------
```

```
      USING phrase
      OUTPUT PROCEDURE phrase
         THRU
      GIVING phrase
   The SORT statement (multiple SORT statements are
         permitted). . . . . . . . . . . . . . . . . . . . .  VII-14
      COLLATING SEQUENCE phrase
```

2.12.4
```
---------------------------------------------------------------------
* The SORT COLLATING SEQUENCE phrase is not implemented.
*         (ST208.A.2)
---------------------------------------------------------------------
```

2.12.5
```
---------------------------------------------------------------------
* The SORT USING file-name-series statement is not implemented.
*         (ST208.A.3)
---------------------------------------------------------------------
```

2.13 REPORT WRITER LEVEL 1

2.14 SEGMENTATION LEVEL 1

```
        Language Concepts
           User-defined words. . . . . . . . . . . . . . . . . . . . .  I-76
              segment-number

        Procedure Division
           Segment-numbers . . . . . . . . . . . . . . . . . . . . . . IX-4
              Fixed segment-number range 0 through 49
              Non-fixed segment-number range 50 through 99
              All sections with the same segment-number must
                 be together in the source program
2.14.1 -----------------------------------------------------------------------
       * Independent segments are not provided in their initial states.
       *         (SG102.B.1, SG103.B.1, SG201.B.1, SG203.B.1)
       -----------------------------------------------------------------------
```

2.15 SEGMENTATION LEVEL 2

All elements of 1 SEG 0,2 are a part of 2 SEG 0,2

Environment Division
   The OBJECT-COMPUTER paragraph
      SEGMENT-LIMIT. IX-5

Procedure Division
   Segment-numbers IX-4
      Sections with the same segment-number need not
         be physically contiguous in the source program

2.16 LIBRARY LEVEL 1

2.16.1 ---

* The Library Module Level 1 is not implemented.

Language Concepts	
User-defined words	I-76
text-name	
All divisions	
The COPY statement	X-2

2.17 LIBRARY LEVEL 2

2.17.1 ---

* The Library Module Level 2 is not implemented.

All elements of 1 LIB 0,2 are a part of 2 LIB 0,2

Language Concepts	
User-defined words	I-76
library-name	
All divisions	
The COPY statement	X-2
OF library-name	
REPLACING phrase	

2.18 DEBUG LEVEL 1

2.18.1 --

* The Debug Module Level 1 is not implemented.

2.19 DEBUG LEVEL 2

2.19.1 ---

* The Debug Module Level 2 is not implemented.

All elements of 1 DEB 0,2 are a part of 2 DEB 0,2 .

```
Procedure Division
   USE FOR DEBUGGING statement. . . . . . . . . . . . . . .  XI-4
      ALL REFERENCES OF identifier series
      file-name series
      cd-name series
```

2.20 INTER-PROGRAM COMMUNICATIONS LEVEL 1

2.20.1 ---

* The Inter-Program Communications Module Level 1 is not implemented.

2.21 INTER-PROGRAM COMMUNICATIONS LEVEL 2

2.21.1 --

* The Inter-Program Communications Module Level 2 is not implemented.

All elements of 1 IPC 0,2 are a part of 2 IPC 0,2

```
Procedure Division
  The CALL statement. . . . . . . . . . . . . . . . . . .   XII-5
    identifier
    ON OVERFLOW phrase
  The CANCEL statement. . . . . . . . . . . . . . . . . .   XII-7
```

2.22 COMMUNICATION LEVEL 1

```
---------------------------------------------------------------
*   The COMMUNICATION Module is not currently evaluated as
*   part of an official validation.  See Section 1.9.3.
---------------------------------------------------------------
```

```
Language Concepts
   User-defined words. . . . . . . . . . . . . . . . . . .   I-76
      cd-name
Data Division
   Communication Section . . . . . . . . . . . . . . . . .  XIII-2
   The communication description entry . . . . . . . . . .  XIII-3
      FOR INPUT clause
         END KEY
         MESSAGE COUNT
         MESSAGE DATE
         MESSAGE TIME
         SYMBOLIC QUEUE
         SYMBOLIC SOURCE
         SYMBOLIC SUB-QUEUE-n
         STATUS KEY
         TEXT LENGTH
      FOR OUTPUT clause
         DESTINATION COUNT
         DESTINATION TABLE
            INDEXED BY
         ERROR KEY
         SYMBOLIC DESTINATION
         STATUS KEY
         TEXT LENGTH
Procedure Division
   The ACCEPT MESSAGE COUNT statement. . . . . . . . . . .  XIII-12
   The DISABLE statement . . . . . . . . . . . . . . . . .  XIII-13
      INPUT
      OUTPUT
      KEY identifier/literal
   The ENABLE statement. . . . . . . . . . . . . . . . . .  XIII-15
      INPUT
      OUTPUT
      KEY identifier/literal
   The RECEIVE statement . . . . . . . . . . . . . . . . .  XIII-17
      MESSAGE
      INTO identifier
      NO DATA phrase
   The SEND statement. . . . . . . . . . . . . . . . . . .  XIII-20
      FROM identifier-1 WITH
      WITH EMI
      WITH EGI
      BEFORE/AFTER ADVANCING
         identifier-3 LINES
         integer LINES
         mnemonic-name
         PAGE
```

2.23 COMMUNICATION LEVEL 2

```
---------------------------------------------------------------
*    The COMMUNICATION Module is not currently evaluated as
*    part of an official validation.  See Section 1.9.3.
---------------------------------------------------------------
```

All elements of 1 COM 0,2 are a part of 2 COM 0,2

```
Communication Section
   The communication description entry. . . . . . . . . .  XIII-3
      FOR INPUT
         INITIAL

Procedure Division
   The DISABLE statement. . . . . . . . . . . . . . . . .  XIII-13
      INPUT
         TERMINAL
   The ENABLE statement . . . . . . . . . . . . . . . . .  XIII-15
      INPUT
         TERMINAL
   The RECEIVE statement. . . . . . . . . . . . . . . . .  XIII-17
      SEGMENT
   The SEND statement . . . . . . . . . . . . . . . . . .  XIII-20
      FROM identifier-1
      WITH identifier-2
      WITH ESI
```

SECTION 3. COMPILER STATUS

3.1 Federal Standard COBOL

Section 1.5 explains the four levels of Federal Standard COBOL and their relation to American National Standard COBOL. This section lists the discrepancies described in Section 2 by the Federal level in which the problem occurs. All errors listed for a lower level are also errors in any higher level, even though they are listed only in the lower level. The paragraph number from Section 2 is used to reference the errors in each Federal level.

3.1.1 Low Level

2.1.1	Null SECTIONs are not allowed.
2.1.2	Data items with PIC S9P(17) are not recognized as integers.
2.1.3	SIGN IS TRAILING or SIGN TRAILING SEPARATE CHARACTER clauses are not implemented.
2.1.4	Operational signs are not stripped before a comparison of nonnumeric operands is made.
2.1.5	Leading spaces are truncated from data with ACCEPT FROM SPO.
2.3.1	All elementary items require an INDEXED BY phrase if one appears at the group level.
2.5.1	Extra lines are provided with WRITE AFTER ADVANCING statement.

3.1.2 Low-Intermediate Level

2.7.1	The Relative I-O Module Level 1 is not implemented.
2.14.1	Independent segments are not provided in their initial states.
2.16.1	The Library Module Level 1 is not implemented.
2.18.1	The Debug Module Level 1 is not implemented.
2.20.1	The Inter-Program Communication Module Level 1 is not implemented.

3.1.3 High-Intermediate

2.2.1	The Comma and semi-colon are not interchangeable.
2.2.2	Comment entries are not replaced in DATE-COMPILED paragraph.
2.2.3	Figurative constants are not allowed in literals in the Alphabet-name clauses.
2.2.4	Incorrect totals are given for the INSPECT TALLYING statement with multiple BEFORE and AFTER phrases.
2.2.5	There is a problem with the INSPECT TALLYING statement on a signed numeric field.
2.2.6	There is a problem with the receiving field of the DELIMITER IN phrase of the UNSTRING statement.
2.6.1	Extra lines appear between logical pages with the LINAGE clause.
2.6.2	The EXCEPTION PROCEDURE ON EXTEND clause is not implemented.
2.8.1	The Relative I-O Module Level 2 is not implemented.
2.19.1	The Debug Module Level 2 is not implemented.
2.21.1	The Inter-Program Communication Module is not implemented.

3.1.4 High Level

2.10.1	The ALTERNATE RECORD KEY clause and WITH DUPLICATES phrase are not implemented.
2.12.1	The SAME SORT-MERGE AREA clause is not implemented.

2.12.2	The same file cannot be used for both sorting and merging.
2.12.3	The MERGE COLLATING SEQUENCE clause is not implemented.
2.12.4	The SORT COLLATING SEQUENCE clause is not implemented.
2.12.5	The SORT USING file-name-series clause is not implemented.
2.17.1	The Library Module Level 2 is not implemented.

3.2 American National Standard COBOL

Full American National Standard COBOL consists of the entire set of language elements defined in the ANSI COBOL standard (refer to 1.7). It is also the equivalent of high level Federal Standard COBOL plus the Report Writer module. Therefore, this section lists only those discrepancies found while validating the Report Writer Module.

None

SECTION 4. SOFTWARE ENVIRONMENT

The compiler referenced in this document was validated using the software environment described in this section. When using a modification of the described environment, the compiler may or may not continue to conform to the Standard. It should be noted that during the validation process, an attempt is made to validate as many different options as possible.

The use of compiler options, implementor-names in the Environment Division and any form of optimization which is not described in this report could cause the compiler to produce a program that does not perform according to the specifications of Standard COBOL. Only the environment described in this document has been used with this compiler to satisfy the requirements of FIPS PUB 21-1 and FPMR 101-32.1305.1a. (Any deviations which must be corrected as per the referenced FPMR are described in Sections 2 and 3 of this report.)

1. Options or parameters used on the processor call statement for the compiler: The following options/parameters were used during the validation.

Options specified:

```
$ SET ANSI74
      USASI
      LINEINFO
      LEVEL 2

      OPTIMIZE  ( several Nucleus and all the TH200 series
                  routines were run with OPTIMIZE set on )
```

Options defaulted:

See Burroughs COBOL Reference Manuals 5000656 and 5001241

2. Environment Division implementor-names.

Printer destined files	PRINTER
Tape files	TAPE
Sequential Mass-storage files	100 * 300 DISK
Random Access files	100 * 300 DISK
Sort files (SD)	SORT DISK
Switch names	

Not supported by the compiler.

Source Computer names	B6700/B7700
Object Computer names	B6700/B7700

3. Optimization. The compiler may or may not have optimization features. If optimization is available by option, it was used during the validation process (during a separate execution of the Compiler Validation System) to determine if its use causes the compiler to produce a program which does not give the expected results. If the optimization is invoked through the compiler call statement then it is mentioned in paragraph 1 above. If it is invoked through the introduction of syntax in other than the Data and Procedure Divisions of the source program it is shown below. Optimization which would require modification to the Data and Procedure Divisions is not considered in this report in that it is beyond the scope of the use of standard COBOL and the validation process.

The optimization feature for this compiler is invoked through the compiler call statement. See 1. above. There was no differnce in the execution of the programs when the optimization feature was invoked.

4. Compiler.

Burroughs COBOL Version II.9.1

5. Operating system.

Burroughs MCP II.9.1

SECTION 5. ASCII VALIDATION

5.1 Purpose of ASCII Validation

The ASCII Validation is performed by running a sequence of three CCVS74 programs (SQ118, SQ119, SQ120) using special procedures. The purpose of this special run is to validate that the compiler/operating system being tested is capable of processing ASCII code represented on magnetic tape and punched cards that were produced (in accordance with the appropriate American National Standard) by another system. There is also a magnetic tape and a card file created during the validation which will be taken to another system for further processing. The purpose is to determine whether the compiler/operating system being tested can also produce ASCII representation on magnetic tape and punched cards which can be processed by a another computer system.

5.2 Applicable ANSI Standards

The ASCII Validation is based on several American National Standards and presumes their support by the compiler/operating system being validated. These are:

1. American National Standard Programming Language COBOL X3.23-1974

   - The CODE-SET clause is used to read and write the ASCII files.

   - The PROGRAM COLLATING SEQUENCE clause is used to process the data in ASCII mode as well as native mode.

   - The SIGN...SEPARATE clause is used for signed data and all data is in the DISPLAY (character) mode.

2. American National Standard Code for Information Interchange (ASCII) X3.4-1968. (Note that this describes the code, not the labeling and tape recording formats.)

3. American National Standard Hollerith Punched Card Code, X3.26-1970.

4. American National Standard Magnetic Tape Labels for Information Interchange, X3.27-1969.

5. American National Standard Recorded Magnetic Tape for Information Interchange (800 CPI, NZRI), X3.22-1967.

6. American National Standard Recorded Magnetic Tape for Information Interchange (1600 CPI, PR), X3.39-1973.

The language of the 1974 COBOL Standard provides the capability to accept, process, and produce ASCII code. The ASCII Standard describes the code insofar as the bit arrangement and configuration, but does not address recording techniques, record formats or any labeling scheme. The 800 CPI, NZRI magnetic tape recording standard was used to establish the recording density and techniques. (1600 CPI, PE based on X3.39-1973 "Recorded Magnetic Tape for Information Interchange" could be used under special arrangements.) The tape labeling scheme used in these tests is based on X3.27-1969 but is also compatible with the revision to that tape label standard. Only the VOL1, HDR1, and EOF1 labels are used. The records are fixed length and unblocked.

5.3 ASCII Validation Process

During the validation, the Validation Manager for the Federal COBOL Compiler Testing Service uses the ASCII-encoded magnetic tape and card files in addition to the normal tape files associated with a validation. For the ASCII portion of the validation the following steps are performed:

1. The tape file and card deck (produced on another computer system) are used as input to several programs designed to validate whether the system being validated can accept and process the data as defined by the respective standards. Any changes made during this validation to the source programs reading the data are noted below in 5.4.1.

2. A tape file and card file are produced during the validation which should prove to be identical to the files described in 1 above. These two files are then processed on a different computer system to determine the degree to which the system being validated supports the ASCII standard. Any changes made during this validation to the source program producing the data are noted below in 5.4.2.

5.4 Results for This Validation

5.4.1. The B6700/B7700 systems processed the card deck, the ANSI labeled tape, and the unlabeled tape correctly.

5.4.2. The B6700/B7700 systems produced an unlabeled ASCII tape and card deck which were both verified later as being correct in format and code set.

APPENDIX A

VALIDATION SUMMARY WORKING DOCUMENT

A-1 This appendix is a working paper produced during the validation and documents the results of the compilation and execution of each of the programs comprising the CCVS. The results contained herein are based on the use of the compiler within the Validation Environment identified in this appendix. This appendix (Validation Summary Working Document) is not part of the official Validation Summary Report (VSR) and is not intended to reflect in any way the compiler's usefulness or degree of conformance to the language specifications.

The reader of this appendix should keep in mind that the same problem area may appear in more than one program, but is considered only as one single discrepancy and as such is reflected only once in the body of the VSR. (The VSR will in turn only reference the first occurrence of the problem in the appendix.)

The reference documents for COBOL are American National Standard Programming Language COBOL (X3.23-1974), and Federal Standard COBOL (FIPS PUB 21-1).

VALIDATION ENVIRONMENT

COMPILER IDENTIFICATION:	B6700/B7700 COBOL Version II.9.1
COMPUTER SYSTEM:	Burroughs B6700/B7700
OPERATING SYSTEM:	Burroughs MCP II.9.1

COMMUNICATION LEVEL 1 and LEVEL 2

No Communication programs were run. See Section 1.9.3.

DEBUG LEVEL 1 and LEVEL 2

The Debug module is not implemented for this compiler and the Debug programs were not run.

INTER-PROGRAM COMMUNICATION LEVEL 1 and LEVEL 2

The Inter-Program Communication module is not implemented for this compiler and the Inter-Program Communication programs were not run.

INDEXED I-O LEVEL 1

IX101 through IX107

A. Compilation:

No errors.

B. Execution:

No errors.

INDEXED I-O LEVEL 2

IX201 through IX204

A. Compilation:

No errors.

B. Execution:

No errors.

IX205 through IX208

A. Compilation:

ALTERNATE RECORD KEY clause and its DUPLICATES phrase are not implemented for this compiler. Statements which referenced alternate keys caused fatal diagnostic messages.

B. Execution:

The tests referencing alternate keys had to be deleted. The other tests in programs IX205 and IX206 executed correctly. Programs IX207 and IX208 were not executed.

LIBRARY LEVEL 1 and LEVEL 2

The COBOL 74 Library module is not implemented for this compiler and the Library programs were not run.

NUCLEUS LEVEL 1

NC101 through NC102

A. Compilation:

No errors.

B. Execution:

No errors.

NC103

A. Compilation:

No errors.

B. Execution:

IF-TEST-65 did not execute correctly. This test is a comparison of nonnumeric operands in which one of the operands is a signed numeric elementary item. The operational sign should not participate in the comparison because the numeric operand is treated as though it were moved to an elementary alphanumeric data item of the same size and the contents of this alphanumeric data item were then compared to the nonnumeric operand. (See 5.2.1.1.2 a. Page II-42 and 5.15.4 (4) a. Page II-75).

NC104 through NC108

A. Compilation:

No errors.

B. Execution:

No errors.

NC109

A. Compilation:

No errors.

B. Execution:

1. ACC-TEST-6 accepts 20 characters from the SPO (standard console ACCEPT and DISPLAY device) and compares the characters it has ACCEPTed into an elementary item PIC A(20) with another elementary item PIC A(20). The fields did not compare equal because the leading space was truncated from the characters transmitted from the SPO.

```
Computed result:    ABC          XYZ .
Expected result:     ABC          XYZ .
```

2. ACC-TEST-7 accepts 9 characters from the SPO into an elementary numeric item PIC 9(9) and then compares this item with another elementary numeric item PIC 9(9) value 012345678. The fields did not compare equal. It is important to understand that a total of ten (10) characters were transmitted from the SPO. When the transferred data exceeds the size of the receiving item, only the leftmost characters of the data are stored in the receiving data item.

   See Page II-54 5.4.4 (4) b. The ACCEPT Statement.

NC110 through NC113

A. Compilation:

   No errors.

B. Execution:

   No errors.

NC114

A. Compilation:

1. Null SECTIONs are not allowed by this implementation of the compiler. All instances of a null section had to be deleted.

```
033700* NULL1-NU-L-TEST-13
033800 0 SECTION.
033900 NULL1 SECTION.
034000 NU-L                 SECTION       .
```

2. ELEM-MOVE-TEST-16 and ELEM-MOVE-TEST-17 had to be deleted. The sending item in each of these tests had the following description:

```
014300 01 WRK-DS-LS-1P17-1 PIC S9P(17) SIGN LEADING SEPARATE
014400    CHARACTER VALUE -100000000000000000.
```

   The compiler issued the message SENDING ITEM IS NOT AN INTEGER for both of these tests.

B. Execution:

   IF-EQUAL-TEST-15 did not produce the expected results. This test is a comparison of nonnumeric operands where one of the operands is a signed numeric data item. The operational sign should not participate in the comparison. (See NC103.)

NC115

A. Compilation:

   No errors.

B. Execution:

No errors.

NC116 through NC120

A. Compilation:

The compiler issued the message CONSTRUCT NOT IMPLEMENTED whenever the following descriptions were used:

```
SIGN IS TRAILING SEPARATE
SIGN IS TRAILING SEPARATE CHARACTER
```

All such statements were changed to:

```
SIGN IS TRAILING.
```

B. Execution:

Tests in NC116 which were related directly to SIGN TRAILING SEPARATE CHARACTER failed. The other tests in these programs executed correctly.

NC151 through NC157

A. Compilation:

No errors.

B. Execution:

No errors.

NC158

A. Compilation:

No errors.

B. Execution:

ACC-TEST-6 fails because the leading SPACE is truncated as in NC109.

NC159

A. Compilation:

No errors.

B. Execution:

No errors.

NC160

A. Compilation:

Null SECTIONs and instances of SIGN IS TRAILING SEPARATE CHARACTER caused fatal diagnostics and had to be deleted or changed - same as NC114 and NC116 thru NC120.

B. Execution:

IF-EQUAL-TEST-15 fails. This is the same test as in NC114.

NC161 through NC165

A. Compilation:

No errors.

B. Execution:

No errors.

NUCLEUS LEVEL 2 .

NC201 through NC202

A. Compilation:

No errors.

B. Execution:

No errors.

NC203

A. Compilation:

The comment entries that were not formal comment lines within the DATE-COMPILED paragraph were not replaced by a paragraph of the form:

DATE-COMPILED. current date.

See 2.4 The DATE-COMPILED Paragraph on Page II-4.

B. Execution:

No errors.

NC204

A. Compilation:

No errors.

B. Execution:

ACC-TEST-6 and ACC-TEST-7 fail as in NC109.

NC205 through NC212

A. Compilation:

No errors.

B. Execution:

No errors.

NC213

A. Compilation:

All instances of comma had to be deleted from the DATA DIVISION. Comma in the DATA DIVISION causes the compiler to issue a fatal diagnostic UNRECOGNIZED CONSTRUCT. The comma and semicolon are fully interchangeable whenever they appear as format punctuation characters in various COBOL formats (5.2.1.8 Format Punctuation Page I-73).

B. Execution:

No errors.

NC214

A. Compilation:

The compiler issued the fatal diagnostic UNRECOGNIZED CONSTRUCT in the Alphabet-name clause because of the presence of figurative constants LOW-VALUE, HIGH-VALUE, QUOTE, and SPACES. These figurative constants were replaced by the respective literals:

LOW-VALUE	1
HIGH-VALUE	256
QUOTE	127
SPACES	" "

B. Execution:

No errors.

NC215

A. Compilation:

The compiler issued a fatal diagnostic because comma is not allowed as a separator between Alphabet-name clauses. The comma was deleted.

B. Execution:

No errors.

NC216

A. Compilation:

No errors.

B. Execution:

1. INSPECT-TEST-19 is a test of TALLYING series and REPLACING series as shown below:

```
INSPECT data-name TALLYING count-1 FOR ALL "A"
                           count-2 FOR LEADING "AH"
                           count-3 FOR CHARACTERS BEFORE "."
                           count-4 FOR CHARACTERS AFTER "AL"
                  REPLACING
                           series.
```

Tests 19.02, 19.03, and 19.04 fail with computed and expected results shown below:

19.02	count-2	0 - expected	1 - computed
19.03	count-3	13 - expected	15 - computed
19.04	count-4	5 - expected	6 - computed

Test 19 uses Rules 5.14.4 (6), (8), and (11) on Page II-71.

2. Test 23.02 gave a result of 0 when the expected result was 1 for count-2 in the INSPECT statement shown below:

```
INSPECT signed-numeric-data
    TALLYING count-1 FOR ALL "-"
             count-2 FOR ALL "5".
```

The signed-numeric-data used in the INSPECT statement above was defined as PIC S9(5) VALUE -12345.

Rule 5.14.4 (2) c. describes how the INSPECT functions with signed numeric data - Page II-70. The operational sign was not ignored. Test 23.02 failed because the character "5" was not found.

NC217

A. Compilation:

A SIGN IS TRAILING SEPARATE phrase had to be deleted from the description of a data item used in STRING-TEST-17.01 and 19.01 because the TRAILING SEPARATE construct is not implemented in the compiler.

B. Execution:

. Deletion of the SIGN TRAILING SEPARATE phrase caused tests 17.01 and 19.01 to fail.

NC218

A. Compilation:

No errors.

B. Execution:

Tests UNST-6.02, 7.02, and 9.02 all fail because of a problem with the DELIMITER IN receiving-field phrase of the UNSTRING statement. When the DELIMITED BY phrase has ALL ZERO, ALL "0", or ALL data-name (with value zero) as the actual delimiter; and the string being parsed has more than one contiguous occurrence of zero (000000...), then the DELIMITER IN receiving-field should finally contain a single zero left justified. The remainder of the receiving-field should be space filled. See 5.21.4 Rule (8) Page II-92.

RELATIVE I-O LEVEL 1 and 2

The Relative I-O module is not implemented for this compiler and the Relative programs were not run.

REPORT WRITER LEVEL 1

RW101 through RW104

A. Compilation:

No errors.

B. Execution:

No errors.

SEGMENTATION LEVEL 1

SG101

A. Compilation:

No errors.

B. Execution:

No errors.

SG102

A. Compilation:

No errors.

B. Execution:

SEG-TEST-5 fails because independent segments are not in their initial state each time these segments are made available to the program.

See 2.2.3 Independent Segments Page IX-2.

SG103

A. Compilation:

No errors.

B. Execution:

INITIAL-STATE-TEST-1, FALL-THRU-TEST-6, and GO-TO-ALTER-IND-TEST-7 all fail because independent segments are not provided in their initial states each time they are made available to the program. See SG102.

SG104 through SG106

A. Compilation:

No errors.

B. Execution:

No errors.

SEGMENTATION LEVEL 2

SG201

A. Compilation:

No errors.

B. Execution:

The following tests fail because independent segments are not provided in their initial state:

SEG-TEST-22 thru 37
SEG-TEST-39 thru 43
SEG-TEST-64 THRU 65.

See SG102.

SG202

A. Compilation:

No errors.

B. Execution:

No errors.

SG203

A. Compilation:

No errors.

B. Execution:

The following tests fail because independent segments are not provided in their initial state:

INITIAL STATE PARA-37, 38, 40B, AND 68C.

See SG102.

SG204

A. Compilation:

A null SECTION at line 081100 had to be deleted because it produced a fatal diagnostic.

B.. Execution:

No errors.

SEQUENTIAL I-O LEVEL 1

SQ101

A. Compilation:

No errors.

B. Execution:

WRT-TEST-25 and WRT-TEST-26 both fail. The sequence of WRITE statements for both of these tests is as follows:

```
WRITE AFTER ADVANCING 1
WRITE blank line AFTER ADVANCING 1
WRITE for test-25
WRITE AFTER ADVANCING 1
WRITE blank line AFTER ADVANCING 1
WRITE FROM for test-26
```

In each test, the line in question should appear 2 lines below and one line above the bracketing reference lines. The system produced the lines 1 line below and 2 lines above the reference lines for both test 25 and 26.

SQ102 through SQ121

A. Compilation:

No errors.

B. Execution:

No errors.

SQ151

A. Compilation:

No errors.

B. Execution:

WRT-TEST-25 failed the same as in SQ101. The test line in question was one line below and two above the reference lines instead of two lines below and one above as was expected.

SQ152 through SQ153

A. Compilation:

No errors.

B. Execution:

No errors.

SEQUENTIAL I-O LEVEL 2

SQ201 through SQ212

A. Compilation:

No errors.

B. Execution:

No errors.

SQ213

A. Compilation:

No errors.

B. Execution:

There is a problem with the LINAGE clause. An extra blank line appears before the body of detail lines in WRT-TEST-1, 2, 3, 4, and 5.

This problem appears after the last line of a logical page and the first line on the next logical page.

SQ214

A. Compilation:

No errors.

B. Execution:

WRT-TEST-1 fails with the same extra blank line between logical pages problem as was described for SQ213.

SQ215 through SQ217

A. Compilation:

No errors.

B. Execution:

No errors.

SQ218

A. Compilation:

In the DECLARATIVES SECTION, the EXTEND option had to be deleted as this option is not implemented in this compiler.

B. Execution:

No errors.

SORT-MERGE LEVEL 1

ST101 through ST117

A. Compilation:

No errors.

B. Execution:

No errors.

SORT-MERGE LEVEL 2

ST201

A. Compilation:

A null SECTION had to be deleted at line 086200.

B. Execution:

No errors.

ST202 through ST207

A. Compilation:

No errors.

B. Execution:

No errors.

ST208 through ST212

A. Compilation:

The following constructs are not implemented in this compiler:

1. In the I-O CONTROL paragraph, the SAME SORT-MERGE AREA phrase;

2. In the SORT statement, the COLLATING SEQUENCE clause;

3. In the SORT statement, the SORT USING file name series; and

4. In the MERGE statement, the COLLATING SEQUENCE clause.

B. Execution:

Because of the nature of the tests with regard to the syntax errors above, programs ST208 through ST212 were not executed.

ST213

A. Compilation:

The SAME SORT-MERGE AREA clause was deleted from the I-O-CONTROL paragraph.

B. Execution:

No errors.

ST214

A. Compilation:

The X-27 card had to be changed to ASSIGN the merge file to MERGE DISK. The same file-name cannot be used as a sort file and also as a merge file because of the requirement that a sort file is assigned to SORT DISK and a merge file is assigned to MERGE DISK.

B. Execution:

No errors.

ST215

A. Compilation:

The COLLATING SEQUENCE clause had to be deleted from the MERGE statement.

B. Execution:

This program was not executed.

TABLE HANDLING LEVEL 1

TH101 through TH108

A. Compilation:

No errors.

B. Execution:

No errors.

TH109

A. Compilation:

The compiler requires that every elementary level subordinate to a group item with an INDEXED BY phrase, must also contain an INDEXED BY phrase.

A dummy index had to be added to lines 009700 and 010000 because of fatal diagnostic messages.

B. Execution:

No errors.

TH110

A. Compilation:

No errors.

B. Execution:

No errors.

TH111

A. Compilation:

The SIGN TRAILING SEPARATE CHARACTER had to be changed to SIGN TRAILING because of a fatal diagnostic.

B. Execution:

No errors.

TH151 through TH152

A. Compilation:

No errors.

B. Execution:

No errors.

TABLE HANDLING LEVEL 2

TH201 through TH215

A. Compilation:

No errors.

B. Execution:

No errors.

BIBLIOGRAPHIC DATA SHEET	1. Report No. CCVS74-VSR285	2.	3. Recipient's Accession No.

4. Title and Subtitle

Validation Summary Report #CCVS74-VSR285 (Assigned by Manager of Testing)
Burroughs 6700/7700 II.9.1

5. Report Date 15 February 1978

6.

7. Author(s)

Same as organization - see 9.

8. Performing Organization Rept. No.

9. Performing Organization Name and Address

Federal COBOL Compiler Testing Service
Department of the Navy
Washington, D. C. 20376

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. Sponsoring Organization Name and Address

Automatic Data Processing Equipment Selection Office
Department of the Navy
Washington, D. C. 20376

13. Type of Report & Period Covered

14.

15. Supplementary Notes

16. Abstracts

This Validation Summary Report (VSR) for the Burroughs B6700/B7700 series COBOL Compiler Version II.9.1 (MCP Version II.9.1) provides a consolidated summary of the results obtained from the validation of the subject compiler against the 1974 COBOL Standard (X3.23-1974/FIPS PUB 21-1). The compiler was validated at the High level of FIPS PUB 21-1. The VSR is made up of several sections showing the discrepancies found. These include an overview of the validation which lists all categories of discrepancies by level/module within X3.23-19 , a section relating the categories of discrepancies to each of the Federal levels of the language; and a detailed listing of discrepancies together with the tests which were failed.

17. Key Words and Document Analysis. 17a. Descriptors

Porgramming Languages
Standards
Compilers
COBOL
Verifying
Proving Program Correctness
Software Engineering

17b. Identifiers/Open-Ended Terms

CCVS
CVS

17c. COSATI Field/Group 09/02

18. Availability Statement

Release unlimited

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

19. Security Class (This Report) UNCLASSIFIED

20. Security Class (This Page) UNCLASSIFIED

21. No. of Pages

22. Price